

RDA 工厂区数据结构说明

目录

1. RDA 工厂区数据结构说明	3
1.1 生产信息区域简介	3
1.2 RDA 生产信息数据结构.....	3
1.3 Phase Check 应用实例	5

1. RDA 工厂区数据结构说明

1.1 生产信息区域简介

生产信息区域主要应用于产线生产测试过程中对产品的质量、工艺流程管控，例如存储手机序列号（SN）、流程控制（PhaseCheck）等信息，客户可以根据自己产品和产线的具体情况自定义数据结构。生产信息区域一般是 FLASH 预留的最后一个 Section 或分区。生产信息区域的最大容量一般为 8Kbytes。

1.2 RDA 生产信息数据结构

目前，RDA 0xFAC00200 版本的生产信息数据结构，采用将 8Kbyte Flash 区域划分为 8 个大小相等单元的机制。每个单元的数据结构定义如下：

```
#define TGT_FACTORY_BLOCK_SIZE 1024

#define FACTORY_VERSION                (0xFAC00200)
#define FACTORY_BLOCK_SIZE            (1024)
#define FACT_SIM_COUNT                 (4)
#define FACT_IMEI_LEN                  (9)
#define FACT_SN_LEN                    (64)
#define FACT_MAC_LEN                   (6)
#define FACT_BT_MAC_FLAG               (0X1D0E)
#define FACT_STATION_COUNT             (15)
#define FACT_STATION_NAME_LEN          (12)
#define FACT_STATION_DESC_LEN          (32)
#define FACT_SUPPLEMENTARY_LEN         (544)
#define FACT_DEVICENAME_LEN            (32)
#define FACT_PINCODE_LEN               (16)

typedef UINT8 FACT_IMEI_T[FACT_IMEI_LEN];

typedef UINT8 FACT_SN_T[FACT_SN_LEN];

typedef struct {
    UINT16                                activated;           //0x00000000
    UINT8                                mac[FACT_MAC_LEN]; //0x00000002
} FACT_MAC_T; //Size : 0x8

typedef UINT8 FACT_STATION_NAME_T[FACT_STATION_NAME_LEN];

typedef struct {
```

```

    UINT16 activated;
    UINT8 dev_addr[FACT_MAC_LEN];
    UINT8 dev_name[FACT_DEVICENAME_LEN];
    UINT8 pin_code[FACT_PINCODE_LEN];
} TGT_BT_INFO_T;

typedef struct {
    UINT16 activated;
    UINT8 mac_addr[FACT_MAC_LEN];
    UINT8 mac_ap1[FACT_MAC_LEN];
    UINT8 mac_ap2[FACT_MAC_LEN];
    UINT8 mac_ap3[FACT_MAC_LEN];
    UINT8 reserved[2];
} TGT_WIFI_INFO_T;

typedef struct {
    UINT32 version; //0x00000000
    FACT_IMEI_T imeiSv[FACT_SIM_COUNT]; //0x00000004
    UINT8 mbSn[FACT_SN_LEN]; //0x00000028
    UINT8 mpSn[FACT_SN_LEN]; //0x00000068
    TGT_BT_INFO_T btInfo; //0x000000A8
    TGT_WIFI_INFO_T wifiInfo; //0x000000E0
    FACT_STATION_NAME_T stationNames[FACT_STATION_COUNT];
//0x000000FC
    UINT16 stationPerformed; //0x000001B0
    UINT16 stationPassed; //0x000001B2
    UINT8 stationLastDesc[FACT_STATION_DESC_LEN]; //0x000001B4
    UINT8 supplementary[FACT_SUPPLEMENTARY_LEN]; //0x000001D4
    UINT32 sequence; //0x000003F4
    UINT32 crc; //0x000003F8
    UINT32 crcInverted; //0x000003FC
} FACT_FACTORY_BLOCK_T; //Size : 0x400

```

结构说明：

version: magic number 信息。当结构体改变时，版本信息需进行更新。

imeiSv: 手机 IMEI 号。

mbSn: 手机 bsn 号。

mpSn: 手机 psn 号。

btInfo: 蓝牙设备信息。

wifiInfo: wifi 设备信息。

stationNames: 各站位的名称，最多支持 15 个站位，与 stationPerformed 和 stationPassed 的

Bit0-Bit14 一一对应。即：

stationNames[0] \longleftrightarrow stationPerformed Bit0, stationPassed Bit0
...
stationNames[14] \longleftrightarrow stationPerformed Bit14, stationPassed Bit14

stationPerformed: 提示站位是否已被执行。1 表示已执行。

stationPassed: 提示站位是否通过。1 表示通过，0 表示失败。只有在 stationPerformed 为 1 时，才使用该标志位。

supplementary: 定制信息区。

sequence: 块管理序列号。为差错保护，将 bit[0]位设为 0（否则将被视为无效）。因此，初始序列号为 0，增加步长为 2。

crc: 整个模块的 CRC，crc 本身并不包括在内。从模块的开始，长度为 BLOCK_SIZE-8。

crcInverted: 位反转 CRC，可增强 CRC 校验的鲁棒性。

1.3 Phase Check 应用实例

【假设】总共有四个测试站位，测试顺序分别为 DOWNLOAD、WRITESN、CFT、IMEI。
stationPerformed=0xFFF7, stationPassed=0xFFF3

【分析】stationPerformed 和 stationPassed 的 BIT 位与测试站位的对应关系如下：

BIT0: DOWNLOAD
BIT1: WRITESN
BIT2: CFT
BIT3: IMEI

stationPerformed=0xFFF7，表示当前手机经过了 DOWNLOAD、SN、CFT 测试站，未经过 IMEI 测试站。

stationPassed=0xFFF3，表示当前手机 DOWNLOAD、SN 测试站测试通过，CFT 测试失败。